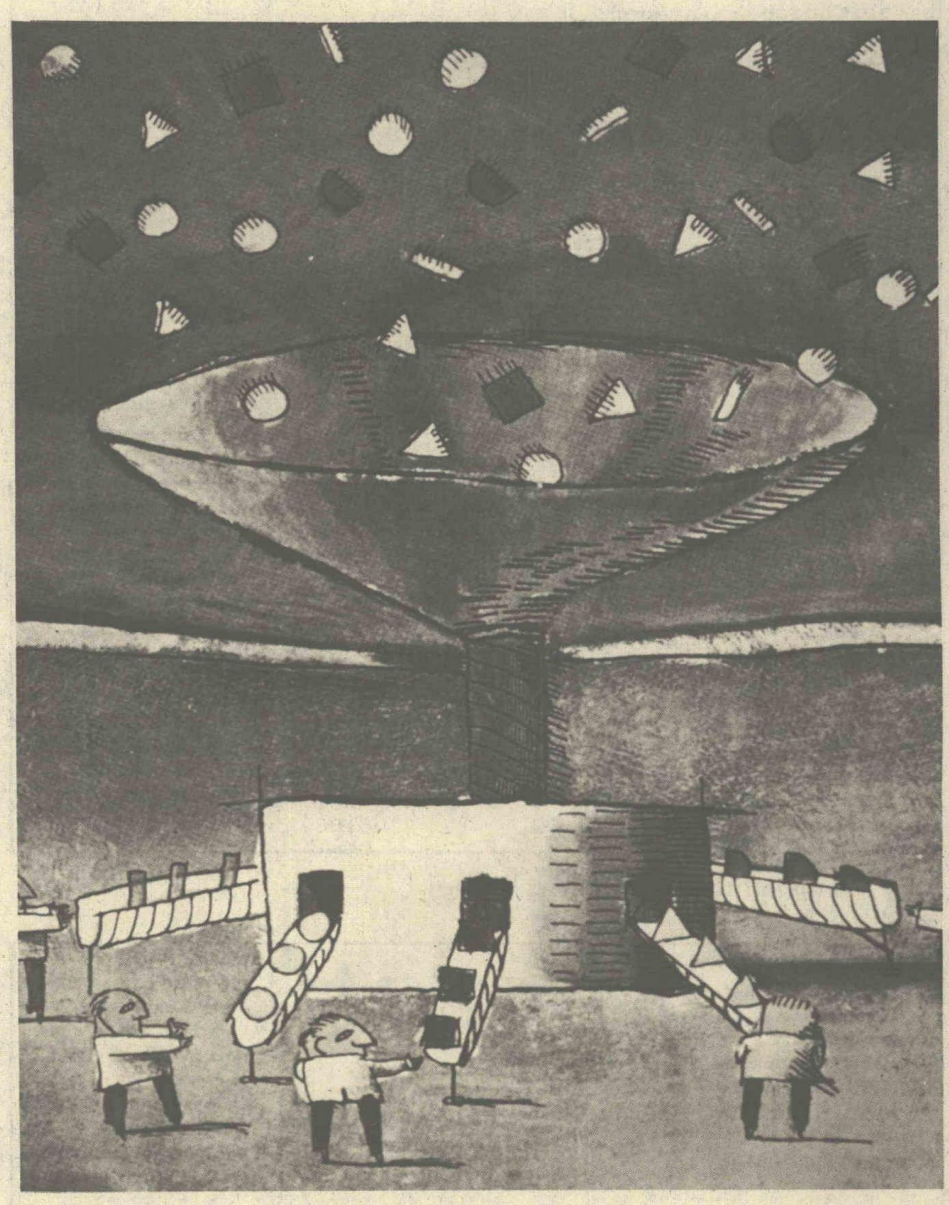
บุญเลิศ เอี่ยมทัศนา

SOFT MAIL

โอโอพี
: แบบกรอกข้อมูล

การเขียนโปรแกรมแบบโอโอพี เพื่อใช้ในการออกแบบ แบบกรอกข้อมูลบนจอภาพ ที่สามารถใช้งานได้ทั่วไป

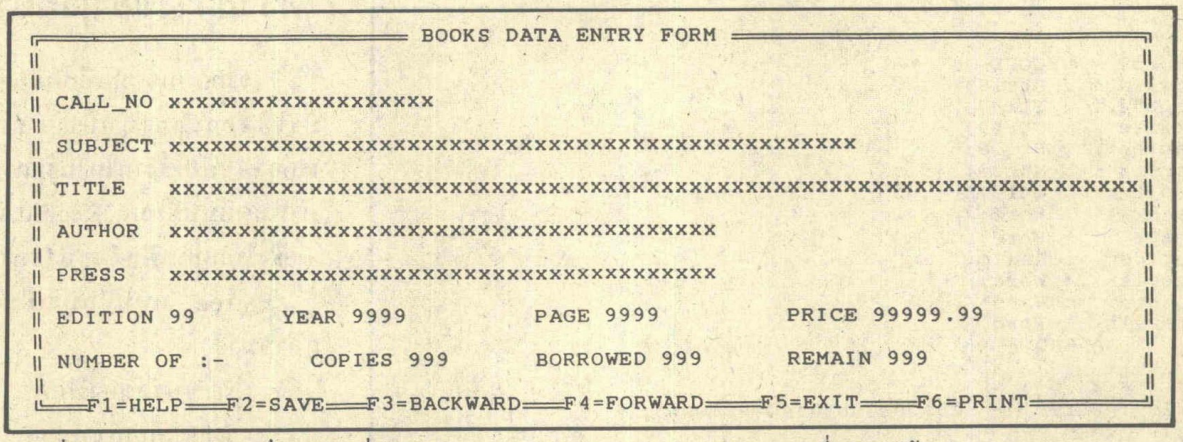


การออกแบบโปรแกรมในแบบโอโอพีคงดำเนินการเช่นเดียวกับแบบเดิม คือเป็นการออกแบบจากบนลงล่าง อย่างเช่นแบบกรอกข้อมูลที่จะนำมากล่าวก็เป็นผลมาจากการออกแบบ แต่การเขียนโปรแกรมในแบบโอโอพีนั้นเพื่อให้ใช้งานได้ทั่วไป (เหนื่อยครั้งเดียว) จึงขอนำมาเสนอแนะและของดแสดงว่าได้มาอย่างไร จะขอกล่าวแต่เพียงว่าจะนำไปใช้งานได้อย่างไร

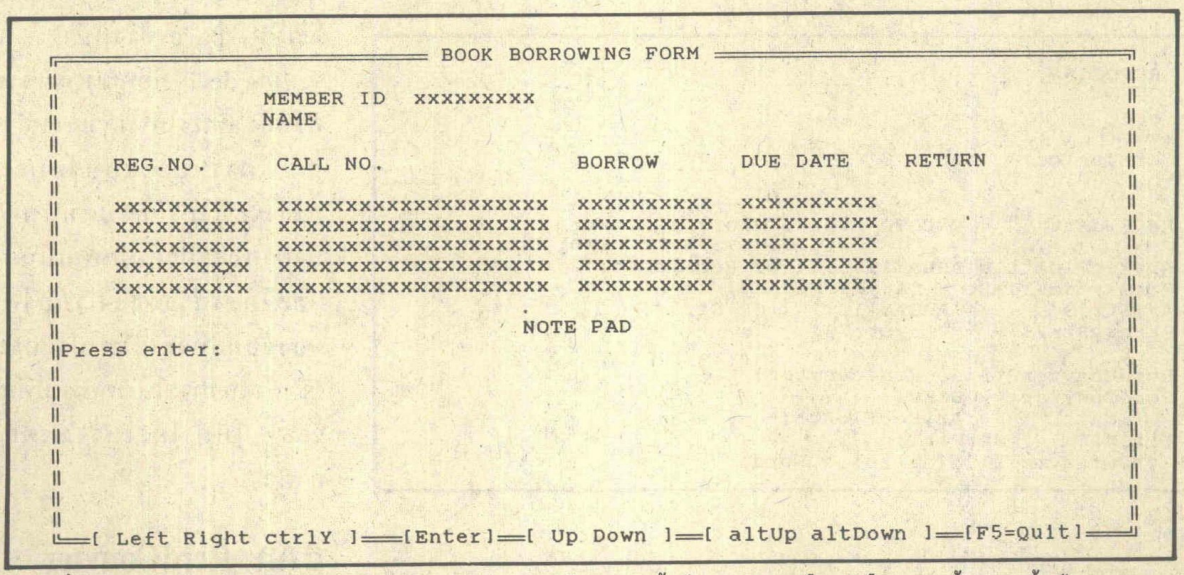
แบบกรอกข้อมูล

การใช้งานคอมพิวเตอร์หนีไม่พ้นการป้อนข้อมูล และเป็นงานที่ใช้เวลาของเครื่องมาก ซอฟต์แวร์ที่ดีจึงอำนวยความสะดวกให้ผู้ใช้ป้อนข้อมูลได้อย่างรวดเร็วผิดพลาดน้อย วิธีการหนึ่งคือทำเป็นแบบกรอกข้อมูลบนจอภาพดังแสดงในรูปที่ 1 และที่ 2 ที่เห็นเป็น xx และ 99 นั้นของจริงจะเป็นเส้นทึบเพื่อให้ผู้ใช้ทราบว่าจะป้อนข้อมูลได้อย่างมากเท่าไร

จากแบบกรอกข้อมูล 2 แบบนี้จะเห็นว่าหากเป็นการเขียนโปรแกรมตามแบบเดิมเป็นไปได้ยากมากที่จะเขียนโปรแกรมแยกกัน เพราะแบบของเรคอร์ดไม่เหมือนกัน แต่เมื่อเขียนโปรแกรมในแบบโอโอพีจะต้องเป็นโปรแกรมเดียวกัน (ไม่เช่นนั้นจะไม่เป็นโอโอพี) และจะต้องให้ใช้งานได้ทั่วไป คือกับเรคอร์ดทุกรูปแบบ (ที่ไม่ซับซ้อนมากนัก)



รูปที่ 1 แบบกรอกข้อมูลรายชื่อหนังสือ ที่เห็นเป็น xx และ 99 คือจำนวนอักษรและ ตัวเลขสูงสุดที่จะกรอกได้



รูปที่ 2 แบบกรอกรายการข้อมูลยืมหนังสือ ฟิลด์ MEMBER ID ป้อนเพียงครั้งเดียวสำหรับการยืมหนังสือ 1 ครั้ง นอกนั้นเป็นการกรอกข้อมูลแบบเป็นตารางบรรทัดละ 1 เรคอร์ด คือหนังสือ 1 เล่ม จะป้อนกี่เล่มก็ได้แต่ไม่เกินจำนวนที่ประมาณไว้

โดยที่แบบกรอกข้อมูลจะทำหน้าที่อ่านข้อมูลที่ผู้ใช้ป้อน (ที่เห็นเป็น xx และ 99) ให้อยู่ในรูปอะเรย์ของสตริง (ฟิลด์ละสตริง) แล้วแปลงเป็นเรคอร์ด หรือแปลงเรคอร์ดเป็นอะเรย์ของสตริง เพื่อนำออกแสดง นั่นคือจะใช้แบบกรอกข้อมูลในการแสดงผลด้วยและจะให้แสดงผลในฟอร์ม (บนจอภาพ) เดิม หรือเปลี่ยนใหม่ได้เช่นกัน

สตริงแบบข้อมูล

จากการที่ต้องการให้ใช้แบบกรอกข้อมูลนี้ได้กับเรคอร์ดทุกแบบ จึงจำเป็นที่ต้องออกแบบข้อมูลใหม่เพื่อใช้กับโปรแกรมนี้ ผมได้พิจารณาดูหลายวิธี เห็นว่า ถ้ากำหนดให้เป็นสตริงจะง่ายที่สุด ถึงแม้จะมีข้อจำกัดอยู่บ้างคือเท่ากับเป็นการจำกัดจำนวนฟิลด์ของเรคอร์ดให้มีได้ไม่เกิน 255 ฟิลด์ (มากพอสมควร) จึงได้เลือกวิธีนี้

ขอให้ดูในรูปที่ 3 ครับ ท่อนบนเป็นการกำหนดแบบข้อมูลเรคอร์ดสำหรับเก็บรายละเอียดของหนังสือ ซึ่งต่อไปไม่จำเป็นต้องกำหนด (ที่นำมาแสดงเพื่อเปรียบเทียบและใช้ในการทดสอบ) ท่อนล่างคือที่กำหนดค่าคงตัว bkfld คือสตริงแบบข้อมูลของเรคอร์ดนี้ที่จะใช้ใน

```
TYPE
  BkRcrdT = RECORD
    callNum : Str19;
    subject : Str49;
    title   : Str69;
    author  : Str39;
    press   : Str39;
    edit    : Word;
    year    : Word;
    page    : Word;
    price   : Real;
    copies  : Word;
    borrow  : Word;
    remain  : Word;
            END;

CONST
  bkfld : Str29 = 's20s50s70s40s40wwwrwww';
```

รูปที่ 3 แสดงการกำหนดสตริงแบบข้อมูล bkfld เทียบกับเรคอร์ดของจริง

```
UNIT  RCRDCONV;

Interface

USES  crt,prime;

TYPE
     RecFldAtt     = RECORD  tpe : Char; si : Word;
                             wid,deci : Byte  END;
     AryRecFldAtt = ARRAY[1..10] OF RecFldAtt;
     PtrAryRecFldAtt = ^AryRecFldAtt;
     AryStr79        = ARRAY[1..10] OF Str79;
     PtrAryStr79     = ^AryStr79;

     RcrdConvterPtr = ^RcrdConvter;
     RcrdConvter = OBJECT
       fldatt   : PtrAryRecFldAtt;
       fldID    : StrPtrT;
       numfld,recsi,fldattsi : Word;
```

รูปที่ 4

โปรแกรม โดยที่ผมกำหนดไว้ว่า

- c หมายถึง Character
- b หมายถึง Byte หรือ Boolean
- i หมายถึง Integer
- w หมายถึง Word
- r หมายถึง Real ซึ่งอาจต่อด้วย : ตัวเลข : ตัวเลข ถ้าไม่มีก็จะให้เอาต์พุตในแบบ scientific notation เช่นเดียวกันกับเมื่อใช้ในคำสั่ง Write และใช้กับข้อมูลแบบอื่นได้เช่นกัน
- s หมายถึง String ซึ่งจะต้องตามด้วยขนาดเป็นตัวเลข เช่น String [19] จะต้องเขียนเป็น S20
- a หมายถึง Array ซึ่งจะต้องตามด้วยขนาดเป็นตัวเลขเช่นเดียวกับสตริง แต่จะไม่มีการดำเนินการใด

นอกจากนี้ยังได้กำหนดอักษรไว้สำหรับกำกับแบบข้อมูลดังกล่าวมาอีก คือ

- x ไม่อ่านหรือเขียนกับฟิลด์นี้
- z ให้นำออกแสดงแต่ไม่ให้เขียนทับฟิลด์นี้

ทั้งนี้เพราะในบางงานอาจมีผู้ใช้หลายคน ซึ่งอาจไม่ต้องการให้ผู้ใช้บางคนเห็นข้อมูล หรือเห็นแต่แก้ไม่ได้ ทั้งโดยเจตนาหรือพลั้งเผลอ

โครงสร้างของเรคอร์ด

จากแบบข้อมูลที่เป็นสตริงดังกล่าวมาจะต้องนำมาทำเป็นโครงสร้างของเรคอร์ด เพื่อใช้งานในโปรแกรม ด้วยการกำหนดแบบข้อมูล RecFldAlt ในรูปที่ 4 ซึ่งในแต่ละฟิลด์จะประกอบด้วย

- tpe แบบข้อมูลเป็นอักขระดังกล่าวมา
- si ขนาดของฟิลด์
- wid คือเลขหลังเครื่องหมาย : ตัวแรก เป็นจำนวนอักขระทั้งหมด (เมื่อแปลงเป็นสตริงแล้ว หากสตริงสั้นกว่าจะเติม space ข้างหน้า)
- deci คือเลขหลังเครื่องหมาย : ตัวหลัง เป็นจำนวนตัวเลขหลังจุดทศนิยม

การแปลงแบบข้อมูลเป็นโครงสร้างจะมีโปรแกรมดำเนินการให้ ที่นำมากล่าวถึงก่อนเพื่อเป็นการแนะนำและโครงสร้างของเรคอร์ดนี้คือแบบของข้อมูลที่จะใช้ในโปรแกรมที่แท้จริง (ซึ่งถ้าต้องการให้เรคอร์ดมีฟิลด์มากกว่า 255 ให้ทำโครงสร้างเรคอร์ดเป็นค่าคงตัว)

สกุล RcrdConvter

ในส่วนของการแปลงข้อมูลไปมาระหว่างเรคอร์ดกับอะเรย์ของสตริง ซึ่งควรจะเป็นกิจกรรมในสกุลของแบบกรอกข้อมูล แต่ผมเห็นว่าน่าจะใช้งานอื่นได้ด้วยจึงทำเป็นสกุลให้ชื่อว่า RcrdConvter และขอนำมากล่าวถึงก่อนคือ

ตัวแปร

ได้กำหนดแบบข้อมูลและตัวแปรเพื่อใช้ในสกุลไว้ดังรูปที่ 4 สำหรับตัวแปรของสกุล คือ

- fldatt เก็บโครงสร้างเรคอร์ดดังกล่าวมา

- fldID เก็บอักษรที่แสดงสถานะของฟิลด์ เรียงลำดับจากฟิลด์แรกถึงฟิลด์สุดท้าย โดยที่ถ้าเป็นฟิลด์ธรรมดาก็จะเป็นอักษรที่เป็นแบบข้อมูล แต่ถ้าเป็นฟิลด์ที่กำกับด้วยอักษร x หรือ z ก็จะเป็นอักษรทั้งสองนี้แทน และต่อไปจะขอเรียกว่า "ฟิลด์ไอดี"

- numfld คือจำนวนฟิลด์ของเรคอร์ด

- fldattsi คือขนาดของหน่วยความจำที่ใช้เก็บโครงสร้างเรคอร์ด (เก็บไว้เพื่อยกเลิกหน่วยความจำนี้)

- recsi เก็บขนาดของเรคอร์ดที่ได้มาจากการแปลงสตริงแบบข้อมูลเป็นโครงสร้างเรคอร์ด

```
    constructor  Init(fieldtype : StrPtrT; numfield : Word);
    procedure    NumToStr(var num,astr; tpe : Char;
                          wid,deci : Byte);

    procedure    StrToNum(var astr,num; tpe : Char;
                          var res : Integer);

    procedure    StrToFld(var astr; rcrd : PtrAryChr;
                          fld : Word; var res : Integer);

    procedure    FldToStr(rcrd,astr : Pointer; fld : Word);
    procedure    ConvRecToStr(inrcrd,outarystr : Pointer);
    procedure    ConvStrToRec(inarystr,outrcrd : Pointer;
                          var res : Integer);

    procedure    ChangeFldID(fld : Byte; newid : Char);
    destructor   Fin;

                      END;

procedure  RecToStruct(fldtpe : StrPtrT; numfld : Word;
                       var fldatt : PtrAryRecFldAtt;
                       var fldID : StrPtrT;
                       var recsi,attsi : Word);
```

รูปที่ 5

กิจกรรม

ได้กำหนดกิจกรรมไว้ดังแสดงในรูปที่ 5 กล่าวคือ

● Init เพื่อเตรียมค่าให้ตัวแปรต่าง ๆ จากการผ่านค่าได้แก่ fieldtype คือแบบข้อมูลที่เป็นสตริงดังกล่าวมา numfield คือจำนวนฟิลด์ของเรคอร์ด โดยเรียกใช้ RecToStruct ดำเนินการให้

● NumToStr

● StrToNum

แปลงข้อมูลเลขจำนวนใน num เป็นสตริงให้แก่ astr และกลับกันตามลำดับ โดยดำเนินการตามแบบข้อมูลใน tpe ประกอบกับ wid และ deci ส่วนค่า res จะเป็นผลมาจากการปฏิบัติตามคำสั่ง Val

● FldToStr

● StrToFld

แปลงเลขจำนวนเป็นสตริงและกลับกัน เช่นเดียวกันและเรียกใช้ NumToStr และ StrToNum ตามลำดับต่างกันเพียงแทนที่จะผ่านค่าแบบข้อมูลเป็นอักขระให้ผ่านเป็นฟิลด์ที่

● ConvRecToStr

● ConvStrToRec

แปลงข้อมูลเรคอร์ดใน inrcrd ทั้งเรคอร์ดเป็นอะเรย์ของสตริงให้แก่ outarystr และกลับกันตามลำดับ โดยเรียกใช้ NumToStr และ StrToNum โดยตรง

● ChangFldID เพื่อการเปลี่ยนค่าไอดี (แบบข้อมูลหรือ x และ z) ในโปรแกรม

● Fin เพื่อยุติการใช้ Rcrd-Convter จะยกเลิกหน่วยความจำของ fldalt และ fldID

โปรแกรมนอกสกุล

ผมได้เขียนโปรแกรมแปลงสตริงแบบข้อมูลเป็นโครงสร้างเรคอร์ดไว้นอกสกุล ซึ่งขัดกับหลักการเขียนโปรแกรมในแบบโอโอพี แค่เห็นว่าในบางงานอาจต้องการโครงสร้างเรคอร์ดโดยไม่ต้องการแปลงข้อมูล จึงเขียนไว้นอกสกุลอยู่ตอนล่างของรูปที่ 2 คือ

● RecToStruct เปลี่ยนสตริงแบบข้อมูลใน fldtpe ตามจำนวนฟิลด์ใน numfld เป็นโครงสร้างเรคอร์ดให้แก่ fldalt และฟิลด์ไอดีให้แก่ fldID และคำนวณขนาดของเรคอร์ดให้แก่ recsi ขอให้สังเกตว่าอาร์กิวเมนต์ของพอยน์เตอร์ทั้งสองตัวนี้เป็นแบบ var ทั้งนี้เพื่อการจองหน่วยความจำในฮีพให้ และให้แจ้งขนาดหน่วยความจำของ

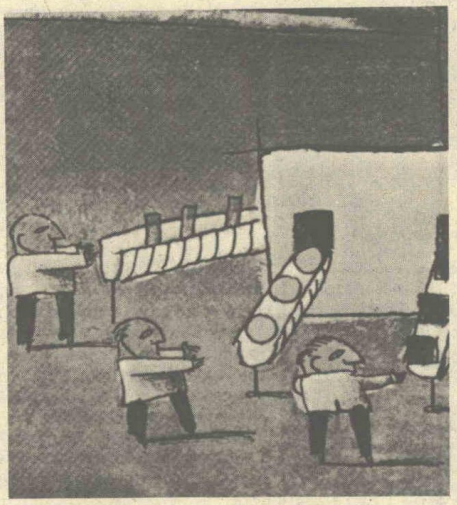
fldalt ทาง altsi (เพื่อการยกเลิก) สำหรับหน่วยความจำของ fldID จะเท่ากับ numfld + 1

คำอธิบายโปรแกรม

โปรแกรมทั้งหมดนี้ (ซึ่งคงต้องพึ่งบริการซอฟต์เมล์ครับ) เขียนอย่างง่าย ๆ ไม่ต้องการคำอธิบาย มีโปรแกรมในเรื่องการแปลงค่าไปมาที่ใช้เทคนิคเฉพาะจึงขอนำโปรแกรม NumToStr มากล่าวถึงดังแสดงในรูปที่ 6 คือ

- ขอให้สังเกตการกำหนดพารามิเตอร์ num กับ astr ว่าไม่มีแบบข้อมูล เทอร์โบปาสคาลให้เราใช้ได้และส่วนมากจะใช้เมื่อไม่ทราบแบบข้อมูลในขณะที่เขียนโปรแกรมเช่นในกรณีนี้

- ในโปรแกรมได้ให้ค่าใน astr ซึ่งทราบว่าเป็นสตริงแน่แก่ stra ด้วย



วิธี type cast เพื่อไม่ต้องใช้แต่ละแห่งในโปรแกรมซึ่งจะทำให้อ่านโปรแกรมยาก แต่ต้องถ่ายค่ากลับไปมา 2 ครั้ง

- ตามที่กล่าวมาว่าเป็นการแปลงค่าเลขจำนวนเป็นสตริงก็เพื่อให้สั้นเท่านั้น ความจริงโปรแกรมนี้จะแปลงหรือถ่ายค่าข้อมูลทุกแบบจาก num ไปให้แก่ astr ซึ่งตอนนี้เราทราบแบบของข้อมูลแล้วจาก tpe คือใน CASE ให้ดำเนินการโดย

- ข้อมูลแบบอักขระ เป็นการถ่ายค่า
- ข้อมูลแบบสตริง เป็นการให้ค่าโดยตรง
- ข้อมูลแบบอะเรย์ เป็นการลอกค่า
- ข้อมูลแบบเลขจำนวนทั้งหมดให้แปลงค่าโดยใช้คำสั่ง Str ร่วมกับ type

```
PROCEDURE  RcrdConvter.NumToStr(var num,astr; tpe : Char;
                                wid,deci : Byte);
VAR  stra : String;
BEGIN
  stra := String(astr);
  case  tpe  of
   'c' : begin  stra[1] := Char(num);  stra[0] := #1  end;
   'b' : Str(Byte(num):wid,stra);
   'i' : Str(Integer(num):wid,stra);
   'w' : Str(Word(num):wid,stra);
   'l' : Str(LongInt(num):wid,stra);
   'r' : Str(Real(num):wid:deci,stra);
   's' : stra := String(num);
   'a' : Move(num,astr,wid);
  end;
  String(astr) := stra;
END;
```

รูปที่ 6

ฟอร์ม

โดยที่ผมนำคำว่า "แบบกรอกข้อมูล" มาใช้เป็นออปเจ็คและเป็นคำรวมด้วยเมื่อจะกล่าวถึงแบบกรอกข้อมูลแท้ ๆ ที่จะปรากฏบนจอภาพจึงต้องเลี่ยงไปใช้คำว่าฟอร์ม ซึ่งมีทั้งเอาต์พุตฟอร์มและอินพุตฟอร์ม ขอให้ดูในรูปที่ 7 ซึ่งเป็นแบบกรอกข้อมูลที่อยู่ (address) จะได้ว่า

เอาต์พุตฟอร์ม

ส่วนที่เป็นเอาต์พุตฟอร์มคือที่แสดงหัวข้อให้ผู้ใช้ป้อน (ซึ่งจะตรงกับชื่อฟิลด์ในเรคอร์ด) แสดงไว้เป็นอักษรใหญ่ และอาจจะมีคำอธิบายหรืออื่น ๆ อีกก็ได้ ได้ทำเป็นแบบข้อมูล Out FrmT

```
                    ADDRESS ENTER FORM

        NAME        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ADDRESS1    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ADDRESS2    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        ADDRESS3    xxxxxxxxxxxxxxxxxx
        ZIP         xxxxxxxxx          PHONE  xxxxxxxxx
F1=HELP  F2=SAVE  F3=BACKWARD  F4=FORWARD  F5=EXIT  F6=PRINT
```

รูปที่ 7
แบบกรอกข้อมูลที่อยู่

ไว้ในรูปที่ 8 สำหรับแต่ละข้อ และทำเป็นอะเรย์ตลอดจนพอยน์เตอร์ของอะเรย์ไว้ด้วย

ที่กำหนดอะเรย์ไว้เป็น 0...10 ไม่ได้หมายความว่าจะจำกัดจำนวนหัวข้อไว้เพียง 11 หัวข้อ เพียงแต่กำหนดให้เป็นอะเรย์ที่เริ่มจาก 0 เท่านั้น จะกำหนดให้เป็น 0...0 ก็ได้ เพราะการใช้งานต่อไปจะใช้พอยน์เตอร์มายังอะเรย์นี้ แต่ละหัวข้อได้กำหนดว่าจะให้แสดงข้อความใน fldnme ที่ใดคือที่โรว์ row และคอลัมน์ col นำไปทำเป็นเอาต์พุตฟอร์มได้ดังรูปที่ 9

อินพุตฟอร์ม

คือส่วนที่แสดงไว้ด้วยอักษร xx เพื่อให้ผู้ใช้ป้อนข้อมูล ได้ทำเป็นแบบข้อมูล InPstnT ตลอดจนทำเป็นอะเรย์และพอยน์เตอร์ไว้ในรูปที่ 8 เช่นกัน เพื่อกำหนดว่าข้อมูลในแต่ละฟิลด์นั้นจะให้อ่านจากโรว์ (row) ใด คอลัมน์ (col) ใดถึงคอลัมน์ใด แต่เพื่อความสะดวกให้ผู้ใช้สามารถบวกค่าความยาวของข้อมูลกับ col ได้โดยตรง จึงกำหนดไว้เป็น aft นำไปทำเป็นอินพุตฟอร์มได้ดังรูปที่ 10

ขอให้ดูการกำหนดแบบข้อมูลในรูปที่ 8 อีกครั้งครับ จะเห็นว่าทั้งสองฟอร์มนี้เป็นอะเรย์ของเรคอร์ด และที่ผมกำหนดให้อะเรย์เริ่มจาก 0 ก็เพราะต้องการให้ใช้ฟิลด์ row ของเรคอร์ดที่ 0 เก็บจำนวนเรคอร์ด (เพื่อลดจำนวนตัวแปรในการผ่านค่า) และสำหรับเอาต์พุตฟอร์มค่าที่ให้แก่ฟิลด์ fldnme ของเรคอร์ดนี้ คือ ADDRESS ENTRY FORM ไม่ได้นำไปใช้

```
UNIT  SMPLFORM;

Interface

USES  crt,prime,smplwnd,rcrdconv;

TYPE
  OutFrmT    = RECORD  row,col : Integer;
                       fldnme : Str29  END;
  AryOutFrmT  = ARRAY[0..10] OF OutFrmT;
  PtrAryOutFrmT   = ^AryOutFrmT;
  InPstnT  = RECORD  row,stt,aft : Integer  END;
  AryInPstnT    = ARRAY[0..10] OF InPstnT;
  PtrAryInPstnT     = ^AryInPstnT;
  AryStr79 = ARRAY[1..10] OF Str79;
  InDataT     = ^AryStr79;
  AryPtrChrs = ARRAY[0..10] OF PtrAryChr;
  PtrAryPtrChrs = ^AryPtrChrs;
```

รูปที่ 8

```
adrOform : array[0..12] of OutFrmT =
 ((row:12; col:00;  fldnme : 'ADDRESS ENTRY FORM'),
  (row:3; col:10;  fldnme : 'NAME'),
  (row:5; col:10;  fldnme : 'ADDRESS1'),
  (row:7; col:10;  fldnme : 'ADDRESS2'),
  (row:9; col:10;  fldnme : 'ADDRESS3'),
  (row:11;col:10;  fldnme : 'ZIP'),
  (row:11;col:40;  fldnme : 'PHONE'),

  (row:13;col:4;   fldnme : 'F1=HELP'),
  (row:13;col:14;  fldnme : 'F2=SAVE'),
  (row:13;col:24;  fldnme : 'F3=BACKWARD'),
  (row:13;col:38;  fldnme : 'F4=FORWARD'),
  (row:13;col:52;  fldnme : 'F5=EXIT'),
  (row:13;col:63;  fldnme : 'F6=PRINT'));
```

รูปที่ 9 เอาต์พุตฟอร์มเพื่อให้ป้อนที่อยู่

```
adrIform : array[0..6] of InPstnT =
  ((row:6; stt:0; aft:0),
   (row:3; stt:21;aft:60),(row:5; stt:21;aft:50),
   (row:7; stt:21;aft:50),(row:9; stt:21;aft:40),
   (row:11;stt:21;aft:30),(row:11;stt:47;aft:56));
```

รูปที่ 10 อินพุตฟอร์มสำหรับอ่านที่อยู่

สกุล SimForm

ถึงตรงนี้ก็พร้อมที่จะสร้างสกุลสำหรับออปเจ็คแบบกรอกข้อมูล ซึ่งผมให้ชื่อว่า SimForm ย่อมาจาก Simple-Form เพราะอาจมีแบบกรอกข้อมูลที่ซับซ้อนกว่านี้

กำหนดความต้องการ

ความต้องการที่ผมจะกล่าวต่อไปนี้ได้มาจากการออกแบบในโครงงานหนึ่ง ซึ่งตัดตอนมาคือ

- แบบกรอกข้อมูลหนังสือตามรูปที่ 1 ต้องการแบบกรอกข้อมูลแบบธรรมดา กรอกข้อมูลครั้งละ 1 เรคอร์ด

- แบบกรอกข้อมูลการยืมหนังสือตามรูปที่ 2 ต้องการแบบกรอกข้อมูลที่เป็นแบบตาราง กรอกข้อมูลได้ครั้งละหลายเรคอร์ด คือข้อมูลการยืมของสมาชิกคนหนึ่ง ๆ จะเก็บไว้เป็นจำนวนเรคอร์ดเท่ากับจำนวนหนังสือที่ยืม คือที่ดูเป็นตารางมีชื่อคอลัมน์เป็น REC NO. ถึง RETURN บรรทัดข้างล่างหนึ่ง ๆ คือเรคอร์ดหนึ่ง ๆ

สำหรับแบบกรอกข้อมูลธรรมดาจะมีกิจกรรมให้ผู้เขียนโปรแกรมคนต่อไปใช้งานดังนี้

- แสดงแบบกรอกข้อมูล
- ให้ผู้ใช้ป้อนข้อมูล (ป้อนและแปลงเป็นเรคอร์ด)
- เก็บแบบกรอกข้อมูล (ลบภาพแบบกรอกข้อมูล)

แบบกรอกข้อมูลที่เป็นตาราง มีลักษณะเป็นผู้เรียกใช้แบบกรอกข้อมูลธรรมดา จึงควรจะเป็นผู้สืบสกุลของแบบกรอกข้อมูลธรรมดา ซึ่งเป็นข้อดีของการเขียนโปรแกรมในแบบโอโอพี คือถ้าในโครงงานแรกต้องการแบบกรอกข้อมูลธรรมดาในโครงงานต่อมาต้องการแบบกรอกข้อมูลแบบตาราง ก็สามารถทำเป็นผู้สืบสกุลได้โดยไม่ต้องแตะต้องซอร์สโค้ดเดิม แต่อาจต้องยอมเสียโปรแกรมเก่าบางส่วนด้วยการกำหนดชื่อทับหรือไม่ใช้

สำหรับในกรณีนี้จะต้องพิจารณาว่าจะรวมไว้ในสกุลเดียวกันหรือให้เป็นผู้สืบสกุลกัน ทั้งนี้เพราะถ้ายึดหลักการเขียนโปรแกรมในแบบโอโอพีอย่างแท้จริง เมื่อสร้างสกุลแบบกรอกข้อมูลธรรมดาจะต้องพิจารณาถึงความต้องการใช้งานในขณะนั้นเท่านั้น ห้ามเผื่อสำหรับอนาคต (แต่ผู้เขียนโปรแกรมก็อดไม่ได้ที่จะเผื่อไว้สำหรับอนาคตบ้าง) ซึ่งจะพิจารณาได้จากความต้องการของแบบกรอกข้อมูลตาราง โดยเฉพาะกับของโครงงานที่ยกมาเป็นตัวอย่าง คือ

- เรคอร์ดของการยืมหนังสือแต่ละเล่มประกอบด้วย หมายเลขสมาชิกซึ่งเป็นส่วนประจำ (ป้อนครั้งเดียวต่อการยืมหนังสือหนึ่งครั้ง) กับส่วนที่เป็นตารางดังกล่าวมา ซึ่งสกุลลูกสามารถเลือกดำเนินการในแต่ละฟิลด์ได้ด้วยการเปลี่ยนฟิลด์ไอดี

- เฉพาะการยืมหนังสือ เมื่อผู้ใช้ป้อนหมายเลขสมาชิก (NUMBER ID) กับเมื่อป้อนวันยืม (BORROW) จากการออกแบบโปรแกรมจะให้ไปค้นชื่อสมาชิก (NAME) และคำนวณวันส่งคืน (DUE DATE) นำออกแสดง นั่นคือจะต้องให้กิจกรรมที่ดำเนินการเกี่ยวกับการป้อนข้อมูลใน SimForm เป็น virtual

- ต้องทำให้การแสดงอินพุตฟอร์มเปลี่ยนตำแหน่งที่ได้ คือเปลี่ยนค่า row ในแต่ละบรรทัด งานนี้สามารถดำเนินการได้ในสกุลลูก

จากที่กล่าวมานี้จะเห็นว่าไม่จำเป็นต้องรวม 2 สกุลนี้เข้าด้วยกัน เพราะงานทั้งสองแบบมีพอ ๆ กันจะได้เลือกสกุลใช้ให้เหมาะแก่ความต้องการของงาน และถูกหลักการเขียนโปรแกรมในแบบโอโอพี แต่สำหรับผู้เขียนเห็นว่าการสืบสกุลหลายชั้นทำให้เสียเวลาประกอบกับเทอร์โบปาสคาลกล่าวไว้ว่าจะไม่รวมโปรแกรมย่อยในสกุลที่ไม่ใช้ไว้ในไฟล์ .EXE เช่นเมื่อใช้สกุลนี้ในฐานะแบบกรอกข้อมูลธรรมดา จะไม่รวมโปรแกรมย่อยในส่วนที่เป็นตารางไว้ในไฟล์ .EXE จึงจะรวมสกุลทั้งสองนี้เข้าด้วยกัน

วางแนวความคิด

1. เนื่องจากแบบกรอกข้อมูลมีการอินพุตและเอาต์พุตกับจอภาพในแบบที่ต้องการการควบคุม และแบบกรอกข้อมูลแบบตารางต้องการบานหน้าต่างหลายบาน จึงจะให้แบบกรอกข้อมูลเป็นผู้สืบสกุลของสกุลหน้าต่าง

2. การแปลงข้อมูลจากเรคอร์ดเป็นอะเรย์ของสตริง และกลับกันจะใช้บริการของสกุล RcrdConvter โดยกำหนดเป็นอินสแตนซ์ของสกุลนี้

3. จะให้ตรวจการป้อนข้อมูลที่เป็นเลขจำนวน หากผู้ใช้ป้อนผิดคือมีอักขระที่ไม่ใช่ตัวเลข จะไม่รับการป้อนและแจ้งให้ผู้ใช้ทราบ จึงต้องการพรอมเตอร์โดยจะให้เป็นอินสแตนซ์ของสกุล Prompter

4. การเขียนและการอ่านกับจอภาพจะใช้บริการในสกุล Window นั่นคือผู้ใช้สามารถป้อนข้อมูลในแบบต่อท้าย แทรก หรือทับได้โดยสะดวก

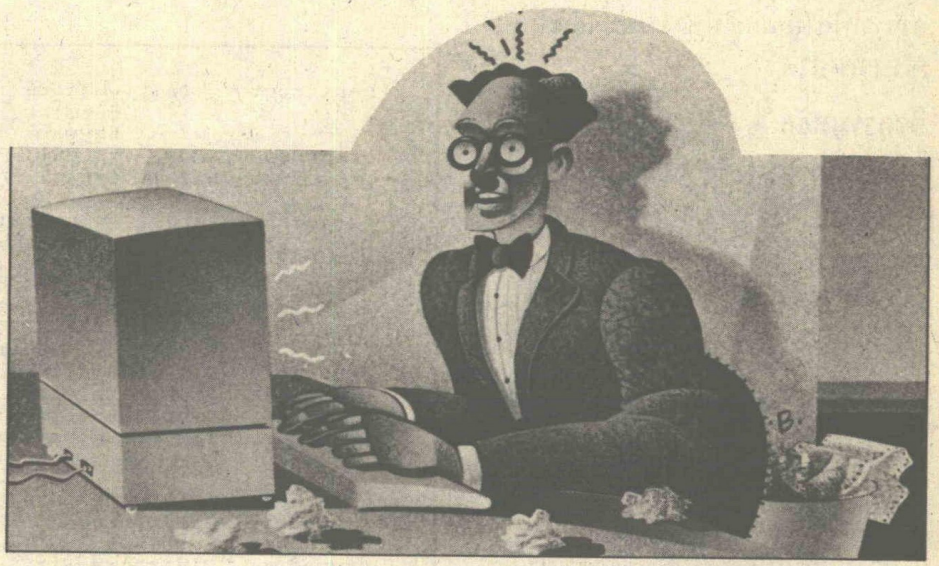
5. ให้ผู้ใช้สามารถเลือกฟิลด์ที่จะป้อนหรือแก้ได้ด้วยคีย์ลูกศรขึ้นและลูกศรลงเมื่อป้อนฟิลด์ใดแล้วให้กดคีย์ Enter จะเลื่อนเคอร์เซอร์ไปต้นฟิลด์ถัดไป แต่ถ้าเป็นฟิลด์สุดท้ายหรือเมื่อกดคีย์ F2 แสดงว่าผู้ใช้ต้องการจบการป้อนข้อมูลของเรคอร์ดนั้น

6. การทำงานเป็นตารางให้เลือกเรคอร์ดได้ด้วยคีย์ PgUp และ PgDn สำหรับคีย์ที่ใช้เพื่อแสดงว่าต้องการจบการป้อนข้อมูลจะให้ผู้เขียนโปรแกรมคนต่อไปเป็นผู้กำหนด

7. การทำงานเป็นตาราง เมื่อผู้ใช้ป้อนเรคอร์ดไปจนถึงบรรทัดสุดท้ายของบานหน้าต่าง หรือเมื่อกดคีย์ PgDn ที่บรรทัดสุดท้ายจะให้เลื่อนเรคอร์ดทั้งหมดขึ้นบน 1 บรรทัด เพื่อให้ป้อน

ข้อมูลเป็นจำนวนเรคอร์ดมากกว่าความสูงของหน้าต่าง และเช่นกันเมื่อกดคีย์ PgUp ที่บรรทัดบนสุดหากไม่ใช่เรคอร์ดแรกจะเลื่อนเรคอร์ดลง 1 บรรทัด

8. ในกรณีที่แบบกรอกข้อมูลมีทั้งส่วนประจำ (ป้อนข้อมูลครั้งเดียว) และส่วนที่เป็นตาราง ซึ่งตามที่กล่าวว่าสามารถแยกจากกันได้ด้วยการเปลี่ยนค่าฟิลด์ไอดีในการทำงานแต่ละส่วน เพื่อเป็นการแบ่งเบาภาระของผู้เขียนโปรแกรมคนต่อไป จะเปลี่ยนเป็นให้กำหนดฟิลด์เริ่มต้นและสุดท้ายในการทำงานแต่ละครั้งแทน



ตัวแปร

จากที่กล่าวมาออปเจ็คของสกุลนี้คือแบบกรอกข้อมูล จึงมีรายละเอียดเป็นอินพุตฟอร์มกับเอาต์พุตฟอร์มเท่านั้น แต่เพื่อสนับสนุนการเขียนโปรแกรมจึงได้กำหนดตัวแปรไว้ดังที่แสดงในรูปที่ 11 คือ

```
SimForm  = OBJECT(SimWnd)
  convter : RcrdConvter;
  formprompt : Prompter;

  data     : PtrAryPtrChrs;
  currcrd  : PtrAryChr;
  oform    : PtrAryOutFrmT;
  iform    : PtrAryInPstnT;
  fldatt   : PtrAryRecFldAtt;
  fldID    : StrPtrT;
  idata    : InDataT;
  numout,numin,fldstt,lstfld  : Integer;
  numrec,lstrec : LongInt;
  recsi    : Word;
  frsttime,formhide : Boolean;
  usrkey,tabwant,fixwant  : ChrSet;
```

รูปที่ 11

- convter และ formprompt เป็นอินสแตนซ์ของสกุล RcrdConvter และ Prompter ตามที่กล่าวมาตามลำดับ

- oform และ iform คือเอาต์พุตฟอร์มและอินพุตฟอร์ม

- fldatt และ fldID คือโครงสร้างเรคอร์ดและฟิลด์ไอดีใน convter นำออกมาไว้ที่นี้ด้วยเพื่อให้ใช้สะดวกเท่านั้น

- idata คืออะเรย์ของสตริงที่ใช้แปลงข้อมูลกับเรคอร์ด

- currcrd คือเรคอร์ดที่ส่งเข้ามาดำเนินการ

- numout และ numin คือจำนวนเรคอร์ดของเอาต์พุตฟอร์มและอินพุตฟอร์มอ่านมาจากต้นอะเรย์ oform และ iform ตามลำดับเพื่อให้ใช้สะดวก

fldstt และ 1stfld คือฟิลด์แรกและฟิลด์สุดท้ายที่ให้ดำเนินการ โดยปกติ fldstt มีค่าเป็น 1 และ 1stfld มีค่าเท่ากับจำนวนฟิลด์

- frsttime เป็นแฟล็กเพื่อแสดงสถานะว่าเรียกใช้กิจกรรม Show เป็นครั้งแรกหรือไม่

- formhide เมื่อเป็นจริงแสดงว่าขณะนั้นไม่ได้แสดงแบบกรอกข้อมูลหากเป็นเท็จหมายถึงกำลังแสดงอยู่

- usrkey เป็นเซตของอักขระหมายถึงฟังก์ชันคีย์ที่ผู้เขียนโปรแกรมคนต่อไปต้องการ คือให้กลับทันทีที่ผู้ใช้กดคีย์เหล่านี้

- tabwant คือ usrkey บวกด้วย PgUp, PgDn และ F2

- fixwant คือ tabwant บวกด้วยคีย์ลูกศรขึ้นและคีย์ลูกศรลง

- data คืออะเรย์ของเรคอร์ดที่ส่งเข้ามาดำเนินการในแบบตาราง

- numrec คือจำนวนเรคอร์ดที่ส่งเข้ามาดำเนินการในแบบตาราง

- lstrec เรคอร์ดสุดท้ายที่ผู้ใช้ป้อนข้อมูลไว้ ในการดำเนินการแบบ

ตารางเพื่อให้ออกไปทำงานอื่นและกลับเข้ามาใหม่ได้

กิจกรรมหลัก

จากความต้องการดังกล่าวมา สามารถนำมากำหนดเป็นกิจกรรมหลักของสกุลดังแสดงในรูปที่ 12 ได้ดังนี้

● **Init**

- เตรียมหน้าต่างด้วย wndform
- เตรียมอินสแตนซ์ convter ด้วยสตริงแบบข้อมูล rectype และจำนวนฟิลด์ที่อ่านค่ามาจาก formin อ่านให้แก่ numin ด้วย
- เตรียมพรอมพ์เตอร์ตามขนาดของพรอมพ์ที่ต้องการ
- จองหน่วยความจำให้แก่ idata และให้ค่าแก่ตัวแปรต่าง ๆ

● **Show** แสดงแบบกรอกข้อมูล โดย

- ให้ SimWnd.Show แสดงหน้าต่าง
- ตรวจแฟล็ก frsttime ถ้าเป็นจริง จะให้เอาต์พุตค่าในเอาต์พุตฟอร์มและแสดงอินพุตฟอร์ม แล้วให้ frsttime เป็นเท็จ
- ครั้งต่อ ๆ ไป SimWnd.Show จะนำภาพที่เก็บไว้ออกแสดงเอง

● **Hide** เก็บภาพแบบกรอกข้อมูล โดยเรียกใช้ SimWnd.Hide

● **EnterFixData** ให้ผู้ใช้ป้อนข้อมูลตามแบบฟอร์ม อ่านมาแปลงค่าให้แก่เรคอร์ด datrcrd ทีละฟิลด์ ซึ่งถ้าให้ olddat เป็นจริง จะนำข้อมูลใน datrcrd ออกแสดงด้วย (ทำให้ใช้ในการป้อน แก้และแสดงผล) คีย์ที่ผู้ใช้กดครั้งสุดท้ายใน ch หากผู้ใช้กดคีย์ธรรมดา หรือใน chf หากผู้ใช้กดคีย์หน้าที่พิเศษ โดย

- ให้ค่า datrcrd แก่ currcrd เพื่อใช้เป็นโกลบอลในสกุลนี้
- ให้สตริงในอะเรย์ idata มีค่าเป็น space ทั้งหมด เพื่อที่จะแปลงค่า (olddat เป็นจริง) หรือไม่ก็ตาม เมื่อให้เอาต์พุตจะเป็นการลบภาพในช่องบรรทัดทั้งหมดหรือหลังข้อความที่เป็นข้อมูล
- การดำเนินการในแต่ละฟิลด์จะตรวจค่า aft ถ้าเป็น 1 (ต้องให้ไว้ในอินพุตฟอร์ม) จะถือว่าเป็นข้อมูลแบบบูลีนให้ GetBoolean ดำเนินการให้ นอกนั้นให้ GetFieldStr เป็นผู้ดำเนินการ
- เมื่อกลับมาจะนำค่าคีย์ใน ch และ chf มาตรวจว่าเป็นคีย์ที่ต้องการเปลี่ยนฟิลด์ (คีย์ลูกศรขึ้น-ลง และ Enter) หรือคีย์ที่ต้องการให้กลับ (Esc และคีย์ใน tabwant) จะดำเนินการตามที่ผู้ใช้ต้องการ

● **DispData** นำข้อมูลใน datrcrd ซึ่งอยู่ในรูปของเรคอร์ดออกแสดงตามแบบฟอร์ม

● **ShowIform** แสดงช่องบรรทัดที่เป็นอินพุตฟอร์มในแบบกลับ (rev) ทำให้เห็นบนจอภาพ หรือแบบปกติ (norm) ทำให้ไม่เห็น ใช้คู่กับ DispData เมื่อต้องการหรือไม่ต้องการให้เป็นตัวดำบนพื้นขาว และใช้ใน TabDataEntry เพื่อให้เห็นว่ากำลังทำงานอยู่ที่บรรทัดใด

● **PrepTabEnter** ก่อนที่จะใช้กิจกรรม TabDataEnter เพื่อการป้อนข้อมูลแบบตารางจะต้องเรียกใช้กิจกรรมนี้ก่อน (ทั้งนี้เพื่อให้ผู้ใช้สามารถออกจากการทำงานใน TabDataEnter ไปชั่วคราว แล้วกลับมาทำงาน ณ จุดเดิม) โดยให้ค่า aryrec แก่ data ค่า nrec แก่ numrec และให้ lstrec เป็นศูนย์

● **TabDataEnter** เพื่อการป้อนข้อมูลในแบบตาราง คือฟอร์มละหลายเรคอร์ด โดยเรียกใช้ EnterFixData อ่านค่าให้แก่ aryrec ครั้งละ 1 เรคอร์ด โดยที่เมื่อกลับมาจะตรวจว่า

```
constructor  Init(wndform : WndParmT;
                  formout,formin,rectype : Pointer;
                  keywant : ChrSet);
procedure  Show; virtual;
procedure  Hide; virtual;

procedure  SetNewFldSttLst(nstt,nlst : Integer);
procedure  StoreField(var astr : Str79;
                      fld : Integer;
                      var chf : Char); virtual;
procedure  EnterFixData(datrcrd : Pointer;
                        olddat : Boolean;
                        var ch,chf : Char);
procedure  DispData(datrcrd : Pointer);
procedure  ShowIform(att : Byte);
procedure  PrepTabEnter(aryrec : Pointer; nrec : LongInt);
procedure  TabDataEnter(var currec : LongInt;
                        var ch,chf : Char);

destructor  Fin;
procedure  PrepOform;
procedure  OutOldData;
procedure  ClearArrayStr;
procedure  GetBoolean(fld,row,col : Integer;
                      var ch,chf : Char);
procedure  GetFieldStr(var astr : Str79;
                       fld,row,stt,stp :
                       Integer; var ch,chf : Char);

             END;
```

รูปที่ 12

- ถ้า ch เป็นคีย์ Esc ให้กลับทันที

- ถ้า ch เป็นคีย์ Enter แสดงว่าจบการป้อนข้อมูลในเรคอร์ดนั้น ให้ป้อนข้อมูลในเรคอร์ดต่อไป ซึ่งถ้าเป็นบรรทัดสุดท้ายของหน้าต่างจะให้เลื่อนเรคอร์ดทั้งหมดขึ้นบน 1 บรรทัด

- ถ้า chf เป็นคีย์ PgUp ให้เลื่อนไปทำงานในเรคอร์ดบรรทัดบน ถ้าเป็น PgDn ให้เลื่อนไปทำงานในเรคอร์ดของบรรทัดล่าง ทั้งนี้เพื่อให้ผู้ใช้สามารถแก้ข้อมูลในเรคอร์ดที่ป้อนแล้วได้

สำหรับค่าของ currec เมื่อเริ่มแรกต้องให้เป็นศูนย์ (อะเรย์ของเรคอร์ดเริ่มจาก 0 เพื่อให้สอดคล้องกับไฟล์) หากผู้ใช้จบการทำงานตามปกติ (เช่นให้กดคีย์ F6 = Quit) หรือเมื่อกดคีย์ Esc จะได้ค่า currec เป็นจำนวนเรคอร์ดที่ป้อน แต่ถ้าผู้ใช้ต้องการผละจากการป้อนข้อมูลชั่วคราว (เช่นให้กดคีย์ F5 = Exit) จะได้ค่า currec เป็นลำดับที่ของเรคอร์ดขณะนั้น (ขณะที่กดคีย์ F5) ดังนั้นเมื่อกลับมาอีกครั้ง currec จะต้องมีค่าเดิม เพื่อให้ผู้ใช้ดำเนินการที่ค้างอยู่ได้

● SetNewFldSttLst เปลี่ยนแปลงค่าใน fldstt และ lstfld เป็นการกำหนดให้ EnterFixData ดำเนินการจากฟิลด์ใดถึงฟิลด์ใด แทนการกำหนดในฟิลด์ไอดีหรือใช้ร่วมกัน

● Fin ยกเลิก SimWnd, converter และ fomprompt กับยกเลิกหน่วยความจำของ idata และเป็นการปิดการใช้งานสกุลนี้ด้วย

โปรแกรมย่อย

เป็นโปรแกรมย่อยภายในของสกุลที่ไม่ต้องการให้ผู้เขียนโปรแกรมคนต่อไปนำไปใช้ ดังที่แสดงการกำหนดส่วนหัวไว้หลัง Fin จึงของดอธิบายด้วย แต่จำเป็นต้องนำโปรแกรมย่อย GetFieldStr มากล่าวถึง เพื่อความเข้าใจทั้งในด้านการอ่านโปรแกรมและการใช้งานสกุลนี้

● **GetFieldStr** เมื่อ EnterFixData ตรวจพบว่าฟิลด์ที่กำลังดำเนินการอยู่ไม่ใช่บูลีนจะเรียกใช้ GetFieldStr ซึ่งจะให้ผู้ใช้ป้อนข้อมูล อ่านค่า แล้วแปลงค่าเป็นฟิลด์ของเรคอร์ดโดยตรง คือครั้งละฟิลด์ ทั้งนี้เพราะต้องการตรวจสอบว่าผู้ใช้ป้อนเลขจำนวนถูกต้องหรือไม่ ซึ่งต้องแปลงเป็นฟิลด์อยู่แล้วจึงให้เก็บลงเรคอร์ดและให้ดำเนินการกับสตริงด้วยเพื่อไม่ให้มีงานซ้ำซ้อน มีรายละเอียดของการดำเนินการดังนี้

- ค่า astr ที่ EnterFixData ส่งมาให้จะเป็นสตริงว่างก่อนที่จะเริ่มดำเนินการกับฟิลด์แรก

- ตรวจว่า astr เป็นสตริงว่างหรือไม่ถ้าไม่ว่างจะให้เอาต์พุต ดังนั้นในเที่ยวแรกนี้จะไม่ให้เอาต์พุต ด้วยเหตุผลที่จะกล่าวต่อไป

- ให้ผู้ใช้ป้อนข้อมูลด้วย Edit-Scrn1Row (ใน SimWnd)

- ตรวจว่าผู้ใช้กดคีย์อักษร (คือคีย์ที่ไม่ใช่คีย์คอนโทรลหรือฟังก์ชัน) ทั้งนี้เพื่อไม่ต้องแปลงค่า หากผู้ใช้กดคีย์ลูกศรขึ้น-ลงไปมา

- หากเป็นคีย์อักษร หรือ Enter ให้อ่านข้อมูลที่ผู้ใช้ป้อนด้วย InScrnStr (ใน SimWnd) ให้แก่ astr แล้วส่งให้ StoreField ดำเนินการแปลงค่าให้แก่ฟิลด์ของเรคอร์ด

- StoreField จะแจ้งให้ทราบว่าผู้ใช้ป้อนเลขจำนวนถูกต้องหรือไม่ ถ้าไม่ถูกต้องจะให้ป้อนใหม่ และมีเรื่องอื่นอีก

กิจกรรมสนับสนุน

กิจกรรมสนับสนุนของสกุลนี้มีเพียงกิจกรรมเดียวคือ StoreField ซึ่งความจริงเป็นโปรแกรมช่วยภายใน แต่เนื่องจากต้องทำเป็นเวอร์ชวล เพื่อให้ผู้เขียนโปรแกรมคนต่อไปตรวจรับการทำงานในแต่ละฟิลด์ได้ จึงยกเป็นกิจกรรมสนับสนุน มีเรื่องราวที่จะขอเรียนให้ทราบดังนี้

● **StoreField** แปลงสตริงให้เป็นฟิลด์และเก็บในฟิลด์ของเรคอร์ด โดยเรียกใช้ convter.StrToFld ซึ่งจะได้ผลมาด้วยว่าผู้ใช้ป้อนเลขจำนวนถูกต้องหรือไม่ และจะมีการดำเนินการดังนี้

- ถ้าป้อนเลขจำนวนไม่ถูกต้องจะแจ้งให้ผู้ใช้ทราบด้วยตัวพรอมเตอร์ formprompt และให้ค่ารหัส Home แก่ chf

- ถ้าถูกต้องจะตรวจว่าเป็นฟิลด์สุดท้ายหรือไม่ ถ้าเป็นฟิลด์สุดท้ายจะให้ค่ารหัส F2 แก่ chf (จบการป้อนข้อมูลสำหรับเรคอร์ดนี้)

- ให้ astr เป็นสตริงว่าง

เพราะเป็นกิจกรรมที่เป็นเวอร์ชวล ผู้เขียนโปรแกรมคนต่อไปจึงสามารถจับการทำงานในแต่ละฟิลด์ได้ อาจจะทำเพื่อตรวจการป้อนข้อมูลหรือเตรียมการล่วงหน้า เช่น การกรอกข้อมูลในการยืมหนังสือในรูปที่ 9 จะมีการดำเนินการดังนี้ คือ

- หลังจากผู้ใช้ป้อนเลขสมาชิก (MEMBER ID) จะค้นหาชื่อสมาชิกและนำออกแสดงโดยตรงหลัง NAME

- หลังจากผู้ใช้ป้อนเลขเรียกหนังสือ (CALL NO.) คือฟิลด์ที่ 3 แล้วจะอ่านวันที่จากเครื่องให้แก่ astr ดังนั้นเมื่อกลับไปทำงานในฟิลด์ที่ 4 ก็จะแสดง

วันที่ในช่อง BORROW (เพราะ astr ไม่ว่าง)

- หลังจากผู้ใช้รับว่าวันที่ถูกต้อง (กดคีย์ Enter) ในฟิลด์ที่ 4 จะคำนวณกำหนดส่งให้แก่ astr สำหรับแสดงในฟิลด์ที่ 5 ในช่อง DUE DATE

โปรแกรมทดสอบ

หลังจากสร้างสกุลแล้วจะต้องเขียนโปรแกรมขึ้นทดสอบการทำงานของสกุลว่าถูกต้องหรือไม่ แต่เพื่อให้ใช้งานได้จริงด้วย จึงทำเป็นแบบกรอกข้อมูลที่อยู่ (ซึ่งจะทดสอบเฉพาะการทำงานของสกุลนี้ในฐานะแบบกรอกข้อมูลธรรมดาเท่านั้น)

การเตรียมการ

เมื่อจะใช้แบบกรอกข้อมูลจะต้องมีการเตรียมการดังนี้ คือ

1. กำหนดแบบข้อมูลเรคอร์ดเพื่อเป็นแนวทางในการกำหนดสตริงแบบข้อมูล และใช้ในโปรแกรมทดสอบ คือ AdrRcrdI ในรูปที่ 13

2. กำหนดสตริงแบบข้อมูล คือ adrfld ในรูปเดียวกัน

3. กำหนดเอาต์พุตฟอร์ม ดังที่ได้ดำเนินการไว้แล้วในรูปที่ 9

4. กำหนดอินพุตฟอร์มตามที่ดำเนินการไว้แล้วในรูปที่ 10

5. กำหนดค่าพารามิเตอร์ของหน้าต่าง คือ adrwndfrm ในรูปที่ 14

แบบและฟอร์มเหล่านี้ควรป้อนใช้ในไฟล์แยกต่างหากเพื่อ include เข้ากับโปรแกรมทดสอบ ผมป้อนใช้ในไฟล์ ADDRFRM.ICL

ตัวโปรแกรม

ได้เขียนโปรแกรมสำหรับทดสอบไว้ในรูป 15 และ 16 โดยให้ addrform

```
{  ADDRFRM0.ICL  }

TYPE
  AdrRcrdT = RECORD
    name   : Str39;
    addr1  : Str29;
    addr2  : Str29;
    addr3  : Str19;
    ZIP    : Str9;
    phone  : Str9;
             END;

CONST
  adrfld : Str19 = 's40s30s30s20s10s10';
```

รูปที่ 13 แสดงการกำหนดสตริงแบบข้อมูล adrfld เทียบกับเรคอร์ดของจริง

```
adrWndfrm : WndParmT =
   ( name      : 'ADDRESS ENTER FORM';
     pale      : (2,5,78,12,1,1);
     horline   : (none,0,0,0,0,0,0,0,0,0);
     verline   : (none,0,0,0,0,0,0,0,0,0) );
```

รูปที่ 14 พารามิเตอร์ของหน้าต่างใช้กับแบบกรอกข้อมูลที่อยู่

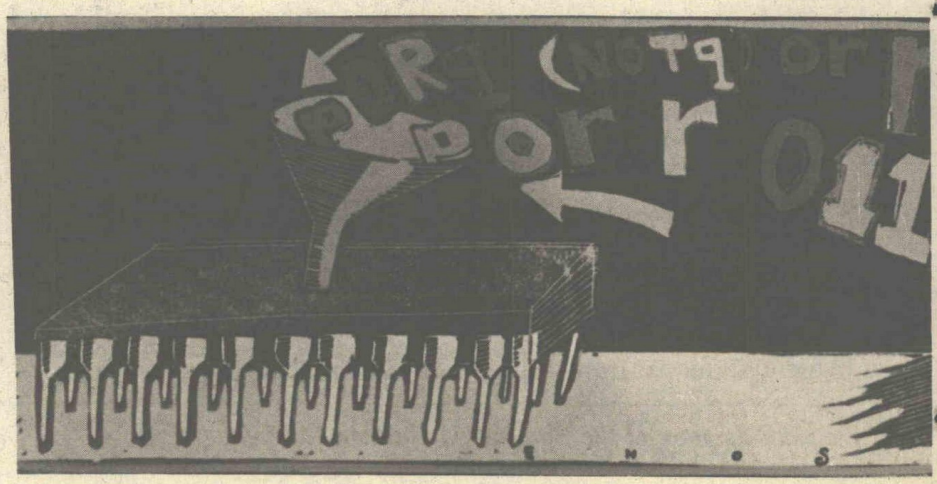
```
PROGRAM  TestFixFormWithAddr;   { TSTFXADR.PAS }

USES  crt,prime,smplwnd,rcrdconv,smplform;

{$I addrfrm0.icl}


VAR
  addrform : SimForm;
  addr : AdrRcrdT;   ch,chf : Char;  si : Word;
  addrfi : FILE OF AdrRcrdT;
  sttmem : LongInt;
```

รูปที่ 15



เป็นอินสแตนซ์ของสกุล SimForm สำหรับตัวโปรแกรมแบ่งออกเป็น 4 ตอนคือ

- ตอนแรก หลังจากเตรียม (Init) addrform แล้ว ให้ตรวจว่าขนาดของเรคอร์ดที่คำนวณได้กับของจริงเท่ากันหรือไม่ ถ้าไม่เท่ากันแสดงว่ากำหนดสตริงแบบข้อมูลคือ adrfld ผิด (ตรวจสอบครั้งแรกครั้งเดียว)

- ตอนที่สอง ให้ผู้ใช้ป้อนข้อมูลครั้งละ 1 เรคอร์ดนำมาเก็บลงไฟล์ จนกว่าผู้ใช้จะกดคีย์ Esc

- ตอนที่สาม นำข้อมูลที่เก็บไว้ในไฟล์ออกแสดงตามแบบฟอร์มที่กำหนดไว้ โดยลบเส้นทึบบนจอภาพด้วย ShowIform และให้งดแสดงเคอร์เซอร์ด้วย CursorOff

- ตอนที่สี่ เปรียบเทียบให้เห็นว่าได้ยกเลิกหน่วยความจำที่จองไว้ในสกุลหมดจดหรือไม่ คือที่ใช้คำสั่ง Fin (สำหรับการตรวจสอบสกุลครั้งแรก)

```
BEGIN   { MAIN }
  sttmem := MemAvail;

  ClrScr;  si := SizeOf(AdrRcrdT);
  addrform.Init(adrWndfrm,@adrOform,@adrIform,@adrfld,[F2]);
  if  addrform.recsi <> si  then
    begin  WriteLn(^G,'mistake in adrfld');  Halt(1)  end;
  addrform.Show;   addrform.InsOnOff;


  Assign(addrfi,'addr0.dta');
  {$I-}  Reset(addrfi);  {$I+}  error := IOresult;
  if  error = 2  then
    begin  ReWrite(addrfi);  Reset(addrfi)  end
  else if  error <> 0  then  Halt(1);
  Seek(addrfi,FileSize(addrfi));
  repeat
    addrform.EnterFixData(@addr,False,ch,chf);
    if  ch <> esc  then  Write(addrfi,addr);
  until  ch = esc;
  Close(addrfi);  Reset(addrfi);

  addrform.ShowIform(norm);  addrform.CursorOff;
  while  not EOF(addrfi)  do
    begin
    Read(addrfi,addr);
    addrform.DispData(@addr);  ReadLn;
    end;
  Close(addrfi);  addrform.CursorOn;
  addrform.Hide;  Window(1,1,80,25);


 Write('PRESS ENTER: ');  ReadLn;  addrform.Fin;
 WriteLn('at start mem = ',sttmem);
 WriteLn('at end   mem = ',MemAvail);
END.
```

รูปที่ 16

ตอนต่อไป

ผมคิดว่าสกุล SimForm มีที่ใช้มากพอสมควรสำหรับการป้อนข้อมูลและการแสดงผลรวมทั้งการแก้ด้วยในงานใดก็ได้ โดยเพียงแต่เขียนโปรแกรมใช้งานสั้น ๆ การออกแบบกรอกข้อมูลบนจอภาพก็ทำได้โดยง่ายโดยเพียงแต่กำหนดเอาต์พุตฟอร์มอินพุตฟอร์ม และรายละเอียดของหน้าต่างเท่านั้น

แต่ถ้าจะให้สมบูรณ์ขอเป็นฉบับหน้าครับที่ผมจะนำเรื่องของแฟ้มข้อมูลมากล่าวถึง เมื่อรวมกันแล้วคุณสามารถที่จะป้อน เก็บ ค้นหา และดึงข้อมูลออกมาใช้ โดยไม่ต้องเขียนโปรแกรมเพื่อดำเนินการในเรื่องต่าง ๆ เหล่านี้อีกเลยกับเรคอร์ดที่ใช้งานกันทั่วไปทุกแบบ

ในซอฟต์เมล์ของฉบับนี้ผมได้ให้โปรแกรมประกอบบทความของผมทั้งในฉบับนี้และฉบับหน้ามาด้วยแล้ว ได้แก่

• ยูนิตสกุล แต่ละไฟล์คือยูนิตหนึ่ง ๆ มีสกุลต่าง ๆ อยู่คือ

- PRIME.PAS กำหนดแบบข้อมูลและค่าคงตัว (ไม่มีสกุล)
- SMPLWND.PAS
- RCRDCONV.PAS
- SMPLFORM.PAS
* DATFI.PAS
* FORMFILE.PAS
* SMPLADR.PAS

ขอให้คอมพายล์แต่ละยูนิตเป็นไฟล์ .TPU (หากใช้ Make หรือ Built แล้วมีปัญหา ควรคอมพายล์ทีละไฟล์)

• โปรแกรมทดสอบ ประกอบด้วยไฟล์ต่าง ๆ ดังต่อไปนี้คือ

- TSTFIX.PAS
- TSTTAB.PAS
* TSTDATFI.PAS
* TSTFRMFI.PAS
* TSTADR1.PAS

หมายเหตุ ไฟล์ที่มีเครื่องหมายดอกจันนำหน้าคือโปรแกรมที่จะใช้ประกอบบทความในฉบับหน้า